STAR Trigger-DAQ Interface Specification Version 1.4

J.Engelage, H. Crawford, L. Greiner, E.G.Judd, M. LeVine, T.Ljubicic, Z.Milosevich, J.Nelson

M.Botlo, V.Lindenstruth, C.McParland, and M.Schultz contributed significantly to previous versions of this document.

Sign-Off Page

The STAR management and group leaders agree to the definition of the interface between the STAR trigger system and data acquisition system as defined in this document. Further changes require the written agreement of all involved parties.

Jay Marx

Richard Jared

Henry Crawford

Mike LeVine

Change Log

November 4, 1994V1.0	First version accepted
November 12, 1997V1.1	modified by H.J.Crawford & J.Nelson
March 13, 1998V1.2	TRG/DAQ disscussion (ej,je,ml,tl,ms attending)
May 20, 1998V1.3	implementation section modified by J.Nelson
26 Oct 1999V1.4	include data formats changes by HJC, JMN, and ZM

1. Introduction

The interface between DAQ and Trigger depends on the architecture which is sketched in Figure 1. All detectors send raw data to their individual DAQ front-end boards. One important aspect of the design is that, due to the space and accessibility constraints of a collider detector, all intelligent components of the DAQ readout chain were moved into the counting house. Consequently the raw data is sent with unidirectional high speed fiber optic links from the detector to the counting house. Trigger actions are signalled to the DAQ front-end systems by state machines on the readout boards of the various detectors using special command words passed from the front-end electronics through these fiber links.

The requirements for the various components shown in figure 1 are listed together with a brief justification in chapter 2. Chapter 3 defines the implementation and the handshake mechanism of the shared memory interfaces between Trigger and DAQ (dark block in figure 1).

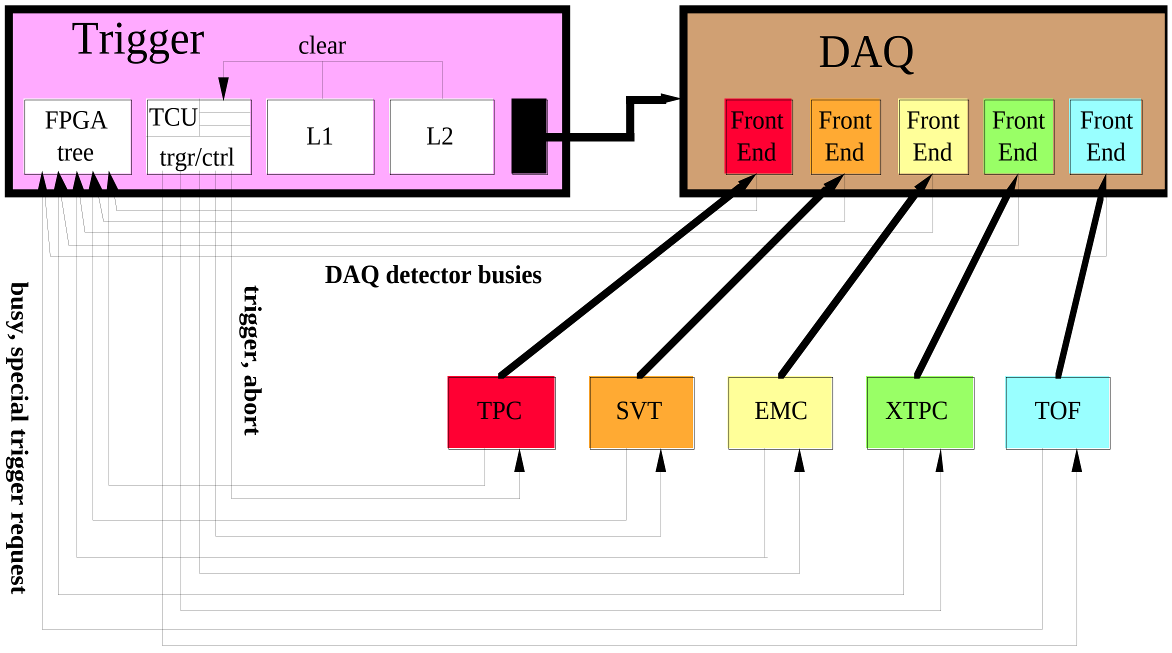


FIGURE 1: *An over all view of the STAR detector, Trigger and DAQ systems. There are other systems like Slow Control and Experiment control that interface with these systems as well. They have been omitted for simplicity. The interface between Trigger and DAQ mainly consists of a shared memory and the DAQ-detector busy signals.*

2. Requirements

2.1 DAQ detector busies

Requirement:

DAQ produces an individual busy signal for each detector system. The logical OR of this signal with the busy of the given detectors front-end electronics forms the detector busy used by the Trigger system.

Justification:

STAR does not have a common dead time. Triggers may involve only one particular detector (like EMC) or a coincidence of several detectors. The DAQ front-end systems implement elasticity buffers that may store a variable number of events. Consequently DAQ front-end boards associated with one particular detector produce a summary busy signal indicating that all internal buffers are filled. The logical OR of these busy signals is the DAQ detector busy which is produced by DAQ and sent to the trigger system as one signal as indicated in figure 1. Another busy signal produced by the detector front-end electronics indicates for example that the given detector electronics is busy digitizing an event. Consequently the trigger system would receive two busy inputs per detector. One for the readout boards on the detector and one for the DAQ front-end in the counting house.

Note:
Since all DAQ front-end boards have their individual detector busies it is possible to read out fast detectors like the EMC while other STAR systems like the TPC are busy. Consequently no additional global DAQ busy is needed for flow control purposes. If for example no data could be read out due to a tape problem the DAQ event builders would simply stop reading out the front-end systems. These systems would assert their busies when their buffers are full preventing any further triggers and consequent data overruns.

2.2 DAQ notification

Requirement:

DAQ is notified by Trigger only if a given event will not be cleared by L1 or L2.

Justification:

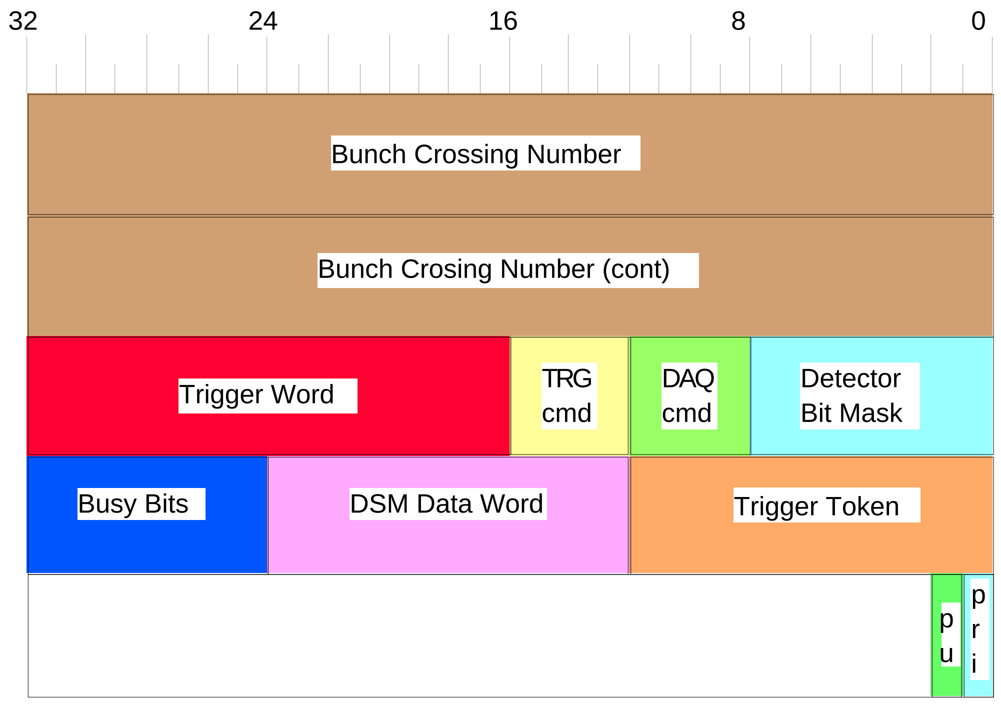
Due to this rule L1 and L2 clears are completely transparent to the DAQ system. No mechanism is required to forward any L1 or L2 aborts to DAQ. This requirement greatly reduces the complexity of the interface between Trigger and DAQ. Its sole functional requirement is to supply DAQ with information about the currently available events for readout.

Note:
The delayed notification of DAQ does not impose any performance burdens since each detector that needs to run at higher rates than the TPC can buffer its TPC related data. It was stated before that the detector readout boards forward clears to the DAQ front-end boards as special commands. However, this does not involve DAQ in a logical sense since the receiving state machines on the front-end boards will be able to perform that clear without any processor intervention.

2.3 Event Descriptor

Requirement

Supply event description of all (L2 accepted) active events. The following words are required in the event descriptor (see Figure 2)



1. Bunch Crossing Number (64 bits)
   The 64 bit crossing number uniquely identifies the time throughout the lifetime of the experiment. Its upper bits are slowly changing and can be derived from the absolute system time.
2. Trigger Action Word[1] (16 bits)
   These bits consists of an 8 bit mask detailing which detectors are participating in the event and must be read out by DAQ, a 4 bit Trigger Command Word[2] instructing each readout board of action to be taken for this event, and a 4bit DAQ Command[3] which defines 16 different DAQ actions related to that event.
3. Trigger Word (16 bits)
   This defines the cause for firing a trigger. It consists of 16 bits which specify the "kind" of event (e.g. high multiplicity, minimum bias scale down event, pedestal event) from the Trigger Word Look Up Table (LUT).
4. Trigger Token (12 bits)
   The trigger token is uniquely linked to an event until it is processed by DAQ. Trigger will not reuse a given trigger token until it has been returned by DAQ.
5. DSM Data Word (12 bits)
   These bits consists of the final 12 bits from the DSM tree.
6. Busy Bits (8 bits)
   These bits consists of the 8 busy bits from the detectors TCD module.
7. Priority Status (1 bit)
   This bit designates which of the two TCU output queues was used for the event, the "priority" event queue (must be read first) or the "normal" event queue.
8. Pile-up Bit (1 bit)
   This "pileup" bit flags TPC events which are preceded or followed by another interaction.

Justification

The trigger system is the only instance within STAR that has complete knowledge about any event. DAQ needs to know which detectors were involved and need to be readout.[4]

1. See Section 3.3.1 of Trigger/Clock Distribution Tree document
2. See Section 2.3 of Trigger/Clock Distribution Tree document
3. See Section 2.5 of Trigger/Clock Distribution Tree document
4. See section 2.3 of Trigger/Clock Distribution Tree document for Trigger Command Definition

2.4 Trigger detectors raw data - Trigger history

Requirement

Supply the complete raw data of the trigger detectors for readout by DAQ including a detector specific pre- and post history. The pre- and post history is limited to 10 bunch crossings for events in coincidence with either the TPC or SVT and 1 bunch crossing for events which involved only *fast* detectors like EMC or TOF.

Justification

The raw data of the trigger detectors that caused the given event to be triggered needs to be in the data stream. The trigger detector data of the N_{pre} previous and N_{post} later events is required to determine pile-up. N_{pre} and N_{post} are configuration constants and depend on the detectors involved. The number of pre and post history events required for the TPC depends on the single track vertex position resolution (1cm). Given the drift velocity of the TPC of about 4cm/μs this corresponds to 250 ns or three bunch crossings. Correspondingly the upper limit of 10 as specified is generous.

Note:
The complete raw data of all trigger detectors will be less than or equal to 2048 bytes in length, since not all the envisioned trigger detectors will be present at first year running. Hence the block length will be allowed to vary from one running period to the next.

2.5 Trigger summary data

Requirement

Supply Trigger summary data of the trigger detectors for further use within L3.

Justification

Results of the L0, L1 and L2 analysis are needed within L3.

2.6 Maximum accepted trigger rate

Requirement

The maximum L2 accepted trigger rate is limited to 1000/sec.

Justification

Limiting the L2 accepted trigger rate simplifies the DAQ event builder design.

Note:

The highest TPC and SVT trigger rate is 100Hz. However, this allows detectors like EMC and TOF to run about an order of magnitude faster if they do not require a coincidence with the TPC.

2.7 Notify DAQ about new events

Requirement

Support asynchronous notification mechanism to DAQ in case of availability of more events.

Justification

The DAQ implementation may require the trigger system to interrupt DAQ upon availability of new events.

2.8 Notify Trigger upon readout completion

Requirement:

Inform Trigger when trigger token is no longer needed and all related data structures in Trigger-DAQ interface are readout.

Justification:

Trigger needs to know when a given event is read out. At that point the trigger token that was associated to the given event is free and can be issued again. All shared data structures linked to the built event are free, too, after the event was completely built.

2.9 Trigger Token

Requirement:

Supply a unique, non zero, 12 bit trigger token to all detectors and DAQ for each event.

Justification:

The STAR detector system will consist of many independent detector systems, which will all have their individual dead time. Any combination of detectors is possible for coincident triggers. The readout time of the various systems varies greatly. Therefore it is possible that during the conversion of one detector another detector is already live and ready to take more data. The STAR trigger system supports firing triggers to fast detectors even if their previously coincident event with a slow detector is still being processed. Therefore it is important to have a mechanism to identify to which event and detector any data buffer belongs. To identify a data buffer with a given event the trigger system will distribute a unique trigger token together with the trigger command itself. One way to make the trigger token unique is to use enough bits to make sure that the same trigger token will never be used again. For example a 64 bit bunch crossing number would be a good example. However, cost considerations require the number of bits transferred per trigger to be as small as possible. Another approach is to make sure that the same trigger token is not used again until the event is assembled. In this scenario the number of bits required for the trigger token is defined by the maximum number of outstanding triggers that are allowed by the data acquisition system. An upper limit for this number is defined by the highest accepted DAQ trigger rate (1000/sec) and the maximum DAQ latency (2s). Correspondingly a 12 bit trigger token would be sufficient. The trigger system will reuse a trigger token only if the corresponding event was readout and assembled by DAQ and the appropriate trigger structures were cleared by DAQ in the Trigger-DAQ interface.

3. Implementation

The interface between the trigger system and DAQ must ensure that data exchange will take place at an average rate of not less than 1kHz.

A dedicated processor, Trigger-DAQ interface (TDI), will reside in the DAQ crate and will receive event descriptors, trigger summary data and trigger detector data from the main trigger crate via an appropriate high-speed network connection (e.g. SCI). This information will be stored in a large memory onboard the TDI CPU.

A communication path will be established between the TDI CPU and the main DAQ processors. This can be accomplished either by VME backplane transfers or some other suitable method.

The trigger token and other data will be exchanged between the TDI and DAQ. Once an event has been fully processed, DAQ will be required to return the trigger token to the TDI which, in turn, will return the token to the main trigger system to signal the completion of the event processing.

If the event was accepted, DAQ will return the token unmodified as the 12 least significant bits in a 32 bit word . If the event was aborted it will return the token with bit 31 (the most significant bit) set. Bit 30 will also be set if the event was rejected at Level 3, or bit 29 will be set if the event was rejected for other reasons.

4. Data Format

An event in trigger consists of an event descriptor, a trigger summary and a block of raw data. There may or may not be additional raw data blocks for the pre and post histroy crossings.

Event Descriptor:

28 byte Event Descriptor Constructed in level 1 and containing all TCU readout

2 bytes byte-count	Length of Event Descriptor including this header (28 B)
1 bytes header	"E" in ASCII for Event Descriptor

1 byte version number	allows up to 256 separate formats for trigger events this has 4 bits for major version and 4 bits for minor
4 bytes data	MS 32 bits of 64-bit Bunch Crossing Number
4 bytes data	LS 32 bits of 64-bit Bunch Crossing Number
2 bytes data	Token
2 bytes data	Action Word:4 bits TRG Cmd 4 bits DAQ Cmd 8 bits Detector Bitmask this bit packing is compiler specific
2 bytes data	Output from last DSM board, saved on TCU
2 bytes data	Address of raw data for this crossing in DSM buffers
1 byte data	Busy bits at start of this Bunch Crossing
1 byte data	Priority and Pileup flags
2 bytes data	Trigger Word
2 bytes data	Number of pre-History raw data blocks - Npre
2 bytes data	Number of post-History raw data blocks - Npost

Trigger Summary

Level 1 and level 2 results are here, along with the data from the DSMs in the TCU crate

2 bytes byte-count	Length of Trigger Summary, including this header (400 B)
2 bytes header	"TS" in ASCII for Trigger Summary
2 bytes byte-count	byte count for DSMs read out for L1 (l = 128 (8 DSMs))
2 bytes header	"L0" in ASCII
l bytes data	Data from DSMs in TCU crate
2 bytes byte-count	byte count for L1 results (m = 128)
2 bytes header	"L1" in ASCII
m bytes data	L1 Results Note: first 4 bytes are used as L1 status for event filters

2 bytes byte-count	byte count for L2 results (n = 128)
2 bytes header	"L2" in ASCII
n bytes data	L2 Results Note: first 4 bytes are for L2 status for event filters

NOTE: the first raw data block is always for the triggered crossing. The data for the Npre crossings comes immediately after that, and the Npost data blocks come last

Raw Data Blocks:

These are the bytes read from the DSMs for each event.

2 byte byte-count:	Length of RAW data block including this header (416 B)
2 byte header:	"RD"
2 byte byte-count:	Length of CTB data block including this header
2 byte header:	"CT"
x bytes data	Data from DSMs in CTB crate (x = 256)
2 byte byte-count:	byte count for MWC and ZDC
2 byte header:	"MZ"
y bytes data	Data from DSMs in MWC/ZDC crate (y = 128)
2 byte byte-count:	byte count for EMC DSMs
2 byte header:	"EC"

------------ below here is just to give an idea

z bytes data	Data from DSMs in EMC crates (z = 32)

----

the RAW data block is repeated here npre and npost times.